2010-04

þÿExtension of Murakami s High nonlinear solver to multiple roots

Neta, B.

http://hdl.handle.net/10945/39444

International Journal of Computer Mathematics
Vol. 87, No. 5, April 2010, 1023–1031

Extension of Murakami's high-order non-linear solver to multiple roots

B. Neta*

Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA, USA

(*Received 05 September 2007; revised version received 20 March 2008; accepted 06 June 2008*)

Several one-parameter families of fourth-order methods for finding multiple zeros of non-linear functions are developed. The methods are based on Murakami's fifth-order method (for simple roots) and they require one evaluation of the function and three evaluations of the derivative. The informational efficiency of the methods is the same as the previously developed methods of lower order. For a double root, the method is more efficient than all previously known schemes. All these methods require the knowledge of multiplicity.

Keywords: multiple roots; non-linear; high-order; efficiency

2000 AMS Subject Classification: 65D99; 41A25

1. Introduction

There is a vast literature on the solution of non-linear equations and non-linear systems, see *e.g.* Ostrowski [12], Traub [18], Neta [10] and references therein. Recently several papers by Sharma [15], Sharma and Goyal [16], Homeier [5] and Grau and Diaz Barrero [2] discuss methods for finding simple roots. Here we develop a high-order fixed point type method to approximate a multiple root. There are several methods for computing a zero ξ of multiplicity m of a non-linear equation $f(x) = 0$ (see Neta [10] and Neta and Johnson [11]. Newton's method is only of first order unless it is modified to gain the second order of convergence (see Rall [13] or Schröder [17]). This modification requires a knowledge of the multiplicity. Traub [18] has suggested to use any method for $f^{(m)}(x)$ or $g(x) = f(x)/f'(x)$. Any such method will require higher derivatives than the corresponding one for simple zeros. Moreover, the first one of those methods require the knowledge of the multiplicity m. In such a case, there are several other methods developed by Hansen and Patrick [4], Victory and Neta [19], Dong [1] and Neta and Johnson [11]. Since in general one does not know the multiplicity, Traub [18] suggested a way to approximate it during the iteration.

*Email: bneta@nps.edu

ISSN 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/00207160802272263
http://www.informaworld.com

For example, the quadratically convergent modified Newton's method is

$$x_{n+1} = x_n - m\frac{f_n}{f_n'} \tag{1}$$

and the cubically convergent Halley's method [3] is

$$x_{n+1} = x_n - \frac{f_n}{((m+1)/2m)f_n' - (f_n f_n'')/(2f_n')}, \tag{2}$$

where $f_n^{(i)}$ is short for $f^{(i)}(x_n)$. Another third-order method was developed by Victory and Neta [19] and based on King's fifth-order method (for simple roots) [8]

$$\begin{aligned} w_n &= x_n - \frac{f_n}{f_n'}, \\ x_{n+1} &= w_n - \frac{f(w_n)}{f_n'}\frac{f_n + Af(w_n)}{f_n + Bf(w_n)}, \end{aligned} \tag{3}$$

where

$$\begin{aligned} A &= \mu^{2m} - \mu^{m+1} \\ B &= -\frac{\mu^m(m-2)(m-1)+1}{(m-1)^2} \end{aligned} \tag{4}$$

and

$$\mu = \frac{m}{m-1}. \tag{5}$$

Yet two other third-order methods developed by Dong [1], both require the same information and both are based on a family of fourth-order methods (for simple roots) due to Jarratt [6]:

$$x_{n+1} = x_n - u_n - \frac{f_n}{(m/(m-1))^{m+1}f'(x_n - u_n) + (m - m^2 - 1/(m-1))^2 f_n'} \tag{6}$$

$$x_{n+1} = x_n - m/(m+1)u_n - \frac{\frac{m}{m+1}f_n}{(1+(1/m))^m f'(x_n - m/(m+1)u_n) - f_n'} \tag{7}$$

where

$$u_n = \frac{f_n}{f_n'}. \tag{8}$$

Neta and Johnson [11] have developed a fourth-order method based on Jarrat's method [7]. The method in general is given by

$$x_{n+1} = x_n - \frac{f_n}{a_1 f_n' + a_2 f'(y_n) + a_3 f'(\eta_n)} \tag{9}$$

where u_n is given by Equation (8) and

$$\begin{aligned} y_n &= x_n - au_n, \\ v_n &= \frac{f_n}{f'(y_n)}, \\ \eta_n &= x_n - bu_n - cv_n, \end{aligned} \tag{10}$$

where the parameters a, b, c, a_1, a_2 and a_3 depend on the multiplicity m.

Our starting point here is Murakami's two-parameter family of methods [9] given by the iteration

$$x_{n+1} = x_n - a_1 u_n - a_2 w_2(x_n) - a_3 w_3(x_n) - \psi(x_n) \tag{11}$$

where u_n is given by Equation (8) and

$$\begin{aligned} w_2(x_n) &= \frac{f_n}{f'(y_n)}, \quad y_n = x_n - a u_n \\ w_3(x_n) &= \frac{f_n}{f'(z_n)}, \quad z_n = x_n - b u_n - c w_2(x_n) \\ \psi(x_n) &= \frac{f_n}{b_1 f'_n + b_2 f'(y_n)}. \end{aligned} \tag{12}$$

Murakami has shown that this family of methods (for simple roots) is of order 5 [9] if the parameters are chosen appropriately. The method requires one function- and three derivative-evaluation per step. Thus the informational efficiency (see [18]) is 1.25.

2. New higher-order scheme

We would like to find the eight parameters a, b, c, a_1, a_2, a_3, b_1 and b_2 so as to maximize the order of covergence to a root ξ of multiplicity m. Let e_n, $\hat{e}_n$, ϵ_n be the errors at the nth step, *i.e.*

$$\begin{aligned} e_n &= x_n - \xi, \\ \hat{e}_n &= y_n - \xi, \\ \epsilon_n &= z_n - \xi. \end{aligned} \tag{13}$$

If we expand $f(x_n)$, and $f'(x_n)$ in Taylor series (truncated after the Nth power, $N > m$) we have

$$f(x_n) = f(x_n - \xi + \xi) = f(\xi + e_n) = \frac{f^{(m)}(\xi)}{m!}\left(e_n^m + \sum_{i=m+1}^{N} A_i e_n^i\right) \tag{14}$$

or

$$f(x_n) = \frac{f^{(m)}(\xi)}{m!} e_n^m \left(1 + \sum_{i=m+1}^{N} B_{i-m} e_n^{i-m}\right) \tag{15}$$

where

$$A_i = \frac{m! f^{(i)}(\xi)}{i! f^{(m)}(\xi)}, \quad i > m \tag{16}$$

$$B_{i-m} = A_i$$

$$f'(x_n) = \frac{f^{(m)}(\xi)}{(m-1)!} e_n^{m-1}\left(1 + \sum_{i=m+1}^{N} \frac{i}{m} B_{i-m} e_n^{i-m}\right). \tag{17}$$

To expand $f'(y_n)$ and $f'(z_n)$ we use some symbolic manipulator, such as Maple [14][1], and find that

$$f'(y_n) = \frac{f^{(m)}(\xi)}{(m-1)!} \hat{e}_n^{m-1}\left(1 + \frac{m+1}{m} B_1 \hat{e}_n + \frac{m+2}{m} B_2 \hat{e}_n^2 + \cdots\right), \tag{18}$$

$$\hat{e}_n = e_n - a u_n = \mu e_n + \frac{a}{m^2} B_1 e_n^2 + \left[\frac{2a}{m^2} B_2 - \frac{a(m+1)}{m^3} B_1^2\right] e_n^3 + \cdots, \tag{19}$$

where

$$\mu = \frac{m-a}{m}. \tag{20}$$

Thus

$$f'(y_n) = \frac{f^{(m)}(\xi)}{(m-1)!} e_n^{m-1} \left(c_0 + c_1 e_n + c_2 e_n^2 + c_3 e_n^3 + \cdots \right), \tag{21}$$

where

$$\begin{aligned}
c_0 &= \mu^{m-1}, \\
c_1 &= \mu^{m-2} \frac{(m-a)^2(m+1) + am(m-1)}{m^3} B_1, \\
c_2 &= \frac{\mu^{m-3}}{m^5} \left[am((m-a)^2 + m(a+1)^2) - \frac{1}{2} a^2 m^2 (m+7) \right] B_1^2 + \frac{\mu^{m-3}}{m^5} [(m+2)(m-a)^4 \\
&\quad + 2a(m-a)m^2(m-1)] B_2, \qquad (22) \\
c_3 &= \mu^m \frac{(m+3)(m-a)^2}{m^3} B_3 - \frac{a\mu^{m-3}}{m^6} [(m^2+3m+2)(a-m)^3 - 2m^2(m+1)(a-m)^2 \\
&\quad - 2m^2 a(m-1)(m-2)] B_1 B_2 + \frac{\mu^{m-4}}{6m^6} a [3(2m^2+4m+1)(a-m)^3 + 2(m+3)(a-m)^2 \\
&\quad + a^2 m^2 (3a + m - 18) + am^2(-3m^2 + 12m + 16) - m^2(5m+6)] B_1^3.
\end{aligned}$$

The error in z_n is given by

$$\begin{aligned}
\epsilon_n = e_n - bu_n - cw_2(x_n) &= \left(1 - \frac{b}{m} - \frac{c}{m} \mu^{1-m} \right) e_n \\
&+ \left(\frac{b}{m^2} + \frac{c}{m^2} \mu^{-m} \left[\mu^2 - \frac{a(1-a)}{m} \right] \right) B_1 e_n^2 + \left[\frac{2b}{m^2} - \left(\frac{\alpha_1}{m^6} c \mu^{-m-1} \right) B_2 \right. \\
&\left. - \left(\frac{\alpha_2}{2m^7} c \mu^{-m-1} + \frac{b(m+1)}{m^3} \right) B_1^2 \right] e_n^3 + \cdots,
\end{aligned} \tag{23}$$

where

$$\begin{aligned}
\alpha_1 &= -a^4(m+2) + 4a^3 m(m+2) - a^2 m^2 (3m+14) + 2m^3(5a - m) \\
\alpha_2 &= 2a^4 m(m+2) - 4a^3 m^2 (m+4) - 12a^3 m + (3m+19) m^3 a^2 + 22a^2 m^2 \\
&\quad - 2am^3(5m+7) + 2a^4 + 2m^5 + 2m^4.
\end{aligned}$$

Now expand $f'(z_n)$ in terms of e_n. To this end, we expand in terms of ϵ_n and substitute for ϵ_n from Equation (23)

$$\begin{aligned}
f'(z_n) &= \frac{f^{(m)}(\xi)}{(m-1)!} \epsilon_n^{m-1} \left(1 + \frac{m+1}{m} B_1 \epsilon_n + \frac{m+2}{m} B_2 \epsilon_n^2 + \cdots \right) \\
&= \frac{f^{(m)}(\xi)}{(m-1)!} e_n^{m-1} \left(d_0 + d_1 e_n + d_2 e_n^2 + d_3 e_n^3 + \cdots \right),
\end{aligned} \tag{24}$$

where

$$
\begin{aligned}
d_0 &= \lambda^{m-1},\\
d_1 &= \lambda^{m-2}\,\frac{\beta_0 + \beta_1 c\mu^{-m} + \beta_2 c^2\mu^{-2m} + \beta_3 bc\mu^{-m}}{m^5}\,B_1,\\
d_2 &= \frac{\lambda^{m-3}}{2m^{10}\mu^{3m+1}}[-(D_1^0 + D_1^1\mu^m + D_1^2\mu^{2m} + D_1^3\mu^{3m})B_1^2\\
&\quad + (D_2^{-1}\mu^{-m} + D_2^0 + D_2^1\mu^m + D_2^2\mu^{2m} + D_2^3\mu^{3m})B_2],
\end{aligned}
\tag{25}
$$

and

$$
\begin{aligned}
\lambda &= \frac{m(m-b) - c(m-a)\mu^{-m}}{m^2},\\
\beta_0 &= (m^2 b(b-m) + m^4\mu^{2m})(m+1),\\
\beta_1 &= m(a^2(m^2-1) + ma(5-m) - m^2(m+3)),\\
\beta_2 &= (m-a)^2(m+1),\\
\beta_3 &= 2m(m+1)(m-a).
\end{aligned}
\tag{26}
$$

The D_i^j are complicated expressions and will be given in the Appendix. Now substitute Equations (15), (17), (21) and (24) into Equations (8), (12) and expand the quotients in Taylor series; then substituting all these into Equation (11), we get

$$
e_{n+1} = C_1^1 e_n + C_2^1 B_1 e_n^2 + (C_3^1 B_1^2 + C_3^2 B_2)e_n^3 + (C_4^1 B_1^3 + C_4^2 B_1 B_2 + C_4^3 B_3)e_n^4 + \cdots, \tag{27}
$$

where the coefficients C_i^j depend on the parameters $a, b, c, b_1, b_2, a_1, a_2$ and a_3. Because of the complexity, we have taken $a = m/2$, $b_2 = 1 - 2^{m-1}b_1$ and either $b = 0$ or $c = 0$. Thus, we reduced the number of parameters to five and as a consequence we were unable to get fifth-order methods. The results for $m = 2$, $m = 3$ and $m = 4$ are given in Table 1.

To summarize, we managed to obtain a family of fourth-order methods requiring one function- and three derivative-evaluation per step. The informational efficiency, $E = p/d$, of these methods

Table 1. The parameters for various values of m.

m	2	2	3	3	4	4
a	1	1	$3/2$	$3/2$	2	2
b	0	Free	0	0.9415780151	0	11.9151259843
c	Free	0	0.2353945038	0	1.9640446368	0
b_1	1	1	free	free	0.05	0.0625
b_2	-1	-1	$1 - 4b_1$	$1 - 4b_1$	0.0268934369	0.5
a_1	-6	-6	$-2.5128989321 - 16b_1$	$-10.571320917 - 16b_1$	-7.49156894	5.6116821612
a_2	3	3	$-1.8238807632 + 4b_1$	$0.1907247330 + 4b_1$	-0.91067191	-1.2089575039
a_3	0	0	4.1469082443	4.1469082443	-0.92646960	-0.4647127230
C_4^1	$15/32$	$15/32$	-6.1027059066	-6.1836740792	-1.35078537	-1.0152077055
C_4^2	$-1/2$	$-1/2$	9.4693139272	9.5300400567	2.141639816	1.5300422793
C_4^3	$1/8$	$1/8$	-3.4826758270	-3.4826758270	-0.822966734	-0.5494657588

Table 2. Comparison of methods for multiple roots.

Method	f	f'	f''	p	d	$E = p/d$	$I = p^{1/d}$
Schröder	1	1	0	2	2	1	1.4142
Hansen and Patrick	1	1	1	3	3	1	1.442
Halley	1	1	1	3	3	1	1.442
Victory and Neta	2	1	0	3	3	1	1.442
Dong	1	2	0	3	3	1	1.442
Neta and Johnson	1	3	0	4	4	1	1.4142
Neta and Johnson, $m = 2$	1	2	0	4	3	1.3333	1.5874
Neta	1	3	0	4	4	1	1.4142
Neta, $m = 2$	1	2	0	4	3	1.3333	1.45874

is 1, as all the aforementioned methods for multiple roots. The efficiency index, $I = p^{1/d}$, is 1.4142 which is lower than the index for those third-order methods. For $m = 2$, we found that $a_3 = 0$ and thus we need one less derivative. This happened also for the methods developed by Neta and Johnson [11]. In this case, the informational efficiency is 4/3 and the efficiency index is 1.5874. Therefore, our method for double roots is more efficient than the Newton's method as modified by Schröder. If the cost of evaluating the first derivative is lower than that of evaluating the function, then our method, for any multiplicity, will be more efficient than the Newton's method (Equation (1)). These results are given in Table 2. Clearly if the cost of evaluating the derivatives is different than that of evaluating the function, one can make an argument to using the appropriate method for the case at hand.

3. Numerical experiments

In all our numerical experiments, we have used the appropriate method with $b = 0$, except for example 3 when we used both schemes. In our first example we took a quadratic polynomial having double roots at $\xi = 1$

$$f(x) = x^2 - 2x + 1. \tag{28}$$

Here we started with $x_0 = 0$ and the root was found in one iteration. The modified Newton method (Equation (1)) converged fast and Newton's method required 10 iterations to get as close as possible to 10^{-7}. In the second example we took a polynomial having two double roots at $\xi = \pm 1$

$$f(x) = x^4 - 2x^2 + 1. \tag{29}$$

Starting at $x_0 = 0.8$ or $x_0 = 0.6$ our method converged in two iterations. The results are given in Table 3.

Similar results were obtained when starting at $x_0 = -0.8$ to converge to $\xi = -1$. For comparison, we have tried the modified Newton. Using $x_0 = 0.6$ we required four iterations to achieve 10^{-9} accuracy.

Table 3. Results for Example 2. $f(x)$ is given by Equation (29).

n	x	f	x	f
0	0.8	0.1296	0.6	0.4096
1	1.00100728	0.4062524998 (−5)	1.03262653	0.004398017
2	1.00000000	0	1.00000036	0.5060180 (−12)

Table 4. Results for Example 3. The first three columns use the scheme with $b = 0$ and the last 3 use $c = 0$. $f(x)$ is given by Equation (30).

n	x	f	n	x	f
0	0	−6	0	0	−6
1	0.989582711	−0.22964188 (−5)	1	0.985370624	−0.64000004 (−5)
2	0.999999994	1 (−18)	2	0.999999974	0

Table 5. Results for Example 4. $f(x)$ is given by Equation (31).

n	x	f	x	f
0	0.1	0.11051709 (−1)	0.2	0.4885611033 (−1)
1	0.2069496569 (−4)	0.428290468 (−9)	0.286951344 (−3)	0.8236470507 (−7)
2	0.43944 (−19)	0.193107514 (−38)	0.162369865 (−14)	0.2636397306 (−29)

The next example is a polynomial with triple root at $\xi = 1$

$$f(x) = x^5 - 8x^4 + 24x^3 - 34x^2 + 23x - 6. \tag{30}$$

The iteration starts with $x_0 = 0$ and the results are summarized in Table 4. The first three columns use the scheme with $b = 0$ and the last three columns use $c = 0$.

Another example with double root at $\xi = 0$ is

$$f(x) = x^2 e^x. \tag{31}$$

Starting at $x_0 = 0.1$ or even $x = 0.2$ our method converged in two iterations. The results are given in Table 5.

The next example having a double root at $\xi = 1$ is

$$f(x) = 3x^4 + 8x^3 - 6x^2 - 24x + 19. \tag{32}$$

Now we started with $x_0 = 0.5$ and the results are summarized in Table 6.

The last example having a root at $\xi = 1$ with multiplicity $m = 4$ and a simple root at $\xi = -1$, *i.e.*

$$f(x) = x^5 - 3x^4 + 2x^3 + 2x^2 - 3x + 1. \tag{33}$$

Now we started with $x_0 = 0.01$ and the results are summarized in Table 7. This is the only case where we needed more than two iterations to converge.

Table 6. Results for Example 5. $f(x)$ is given by Equation (32).

n	x	f
0	0.5	6.6875
1	1.00806166565	0.235014761 (−2)
2	1.00000000024	0.2 (−17)

Table 7. Results for Example 6. $f(x)$ is given by Equation (3).

n	x	f
0	0.01	0.9702019701
1	0.090514708167	0.905147081668 (−1)
2	0.562284899208	0.573490665693 (−1)
3	0.993019776872	0.47313908958 (−8)
4	0.999999999699	0

4. Conclusions

We have extended Murakami's method to obtain non-simple zeros. We have developed a one-parameter family of fourth-order methods for various values of the multiplicity. The methods listed are not the only solution to the system of equations and we only listed a representative scheme. The numerical experiments demonstrate the rapid convergence of our method. Because of the complexity of the symbolic manipulation, we had to assign certain values to some of the parameters and were unable to achieve fifth order.

Note

1. 'Maple is a system for mathematical computation symbolic, numerical and graphical' [14]. For example, the command 'rfp:=series(1/fp1,e,6);' expands the reciprocal of the function $fp1$ into a power series in the variable e keeping terms up to $O(e^6)$. Maple can convert that expansion into a polynomial by using the command 'rfp:=convert(rfp,polynom):'

References

[1] C. Dong, *A family of multipoint iterative functions for finding multiple roots of equations*, Int. J. Comput. Math. 21 (1987), pp. 363 367.
[2] M. Grau and J.L. Diaz-Barrero, *An improvement to Ostrowski root-finding method*, Appl. Math. Comput. 173 (2006), pp. 450 456.
[3] E. Halley, *A new, exact and easy method of finding the roots of equations generally and that without any previous reduction*, Philos. Trans. R. Soc. Lond. 18 (1694), pp. 136 148.
[4] E. Hansen and M. Patrick, *A family of root finding methods*, Numer. Math. 27 (1977), pp. 257 269.
[5] H.H.H. Homeier, *On Newton-type methods with cubic convergence*, J. Comput. Appl. Math. 176 (2005), pp. 425 432.
[6] P. Jarratt, *Some fourth order multipoint methods for solving equations*, Math. Comput. 20 (1966), pp. 434 437.
[7] ———, *Multipoint iterative methods for solving certain equations*, Comput. J. 8 (1966), pp. 398 400.
[8] R.F. King, *A family of fourth order methods for nonlinear equations*, SIAM J. Numer. Amal. 10 (1973), pp. 876 879.
[9] T. Murakami, *Some fifth order multipoint iterative formulae for solving equations*, J. Inform. Process. 1 (1978), pp. 138 139.
[10] B. Neta, *Numerical Methods for the Solution of Equations*, Net-A-Sof, California, 1983.
[11] B. Neta and A.N. Johnson, *High order nonlinear solver for multiple roots*, *Computers and Mathematics with Applications*, accepted for publication, doi:10.1016/j.camwa.2007.09.001.
[12] A.M. Ostrowski, *Solutions of Equations and System of Equations*, Academic Press, New York, 1960.
[13] L.B. Rall, *Convergence of the Newton process to multiple solutions*, Numer. Math. 9 (1966), pp. 23 37.
[14] D. Redfern, *The Maple Handbook*, Springer-Verlag, New York, 1994.
[15] J.R. Sharma, *A composite third order Newton Steffensen method for solving nonlinear equations*, Appl. Math. Comput. 169 (2005), pp. 242 246.
[16] J.R. Sharma and R.K. Goyal, *Fourth-order derivative-free methods for solving non-linear equations*, Int. J. Comput. Math. 83 (2006), pp. 101 106.
[17] E. Schröder, *Über unendlich viele Algorithmen zur Auflösung der Gleichungen*, Math. Ann. 2 (1870), pp. 317 365.
[18] J.F. Traub, *Iterative Methods for the Solution of Equations*, Prentice Hall, New Jersey, 1964.
[19] H.D. Victory and B. Neta, *A higher order method for multiple zeros of nonlinear functions*, Int. J. Comput. Math. 12 (1983), pp. 329 335.

Appendix

Here we list the coefficients D_i^j in the expression for d_2 in Equation (25).

$$\begin{aligned}D_1^0 = &-26c^3a^2m^4 + 32m^4c^3a^3 + 26c^3a^3m^3 - 18c^3a^4m^3 - 28m^5c^3a^2 - 2m^6c^3 - 2m^6c^3a^2\\ &+ 6m^5c^3a^3 - 2m^7c^3 + 2c^3a^5m^3 + 12m^6c^3a - 12c^3a^4m^2 + 4c^3a^5m^2 - 6c^3a^4m^4\\ &+ 12c^3am^5 + 2c^3a^5m.\end{aligned}$$

$$
\begin{aligned}
D_1^1 = {} & -37m^6c^2a - 2m^5c^2a - m^7c^2a - 4m^6c^2a^2b + 7m^6c^2a^3 - m^6c^2a^2 - m^6c^2a^4 \\
& + 72m^5c^2a^2 - 8c^2a^4m^3b + 26m^6bac^2 - 38bc^2a^2m^4 - 4c^2a^4m^2b - m^7c^2a^2 \\
& + 8m^5c^2a^3b + m^4c^2a^5 - 6bc^2m^6 + 22bc^2a^3m^3 - 7m^5c^2a^4 + 2m^4c^2a^2 \\
& + 26bc^2am^5 + m^8c^2 - 42bc^2a^2m^5 - c^2a^5m^3 - 7m^5c^2a^3 + 7m^7c^2 - c^2a^5m^2 \\
& + 9m^4c^2a^4 - 4m^4c^2a^4b + 15c^2a^4m^3 + 30bc^2a^3m^4 + m^5c^2a^5 - 56c^2a^3m^4 - 6m^7c^2b.
\end{aligned}
$$

$$
\begin{aligned}
D_1^2 = {} & -10ca^3m^4b + 2ca^4m^2b - 2m^6ca^2b^2 - 5m^6ca^2 - 8m^3bca^3 - 2m^7c + 2ca^4m^3b \\
& + 6m^7ca^2 + 14m^7bc - 2m^7bca + 2m^5ca^3b^2 + 8m^4bca^2 + 16b^2cam^5 + 14m^6ca \\
& - 2m^8ca + 4m^7ca - m^7ca^2b - 2ca^4m^3 - 2ca^4m^4b + 2m^6ca^4 - 6b^2m^6c - 2m^8c \\
& - 8m^6ca^3 + 2b^2ca^3m^3 - 22ca^2m^5 + 6ca^3m^5 - 2m^7ca^3 + m^8ca^2 + 2m^6ca^3b \\
& - 44m^6bca + 16b^2am^6c + 2m^8bc - 2ca^4m^4 + 4ca^3m^4b^2 + 12ca^3m^4 - 12b^2a^2m^4c \\
& + 37m^5ba^2c - 2m^5ca^4b - 14b^2a^2m^5c - 2m^5bca - 6m^7cb^2 + 4m^6ba^2c + 2ca^4m^5.
\end{aligned}
$$

$$
\begin{aligned}
D_1^3 = {} & 2b^3am^5 - 7m^6b^2a + 2b^3am^6 + 7m^7b^2 + 2m^7ba - 2m^8b + m^8b^2 - 2b^3m^6 \\
& - 2m^7b^3 - 2bm^7 - m^7b^2a + 2bam^6.
\end{aligned}
$$

$$
\begin{aligned}
D_2^{-1} = {} & 20c^4m^4a + 40c^4a^3m^2 - 40c^4a^2m^3 - 20m^4c^4a^2 + 2c^4a^5m + 4c^4a^5 + 10m^5c^4a \\
& - 2m^6c^4 - 10c^4a^4m^2 - 20c^4a^4m + 20c^4a^3m^3 - 4c^4m^5.
\end{aligned}
$$

$$
\begin{aligned}
D_2^0 = {} & -64c^3a^3m^3 - 64c^3am^5 - 16bc^3m^5 + 96c^3a^2m^4 - 8m^6bc^3 + 8c^3a^4m^3 - 32m^6c^3a \\
& - 32m^4c^3a^3 + 16c^3a^4m^2 + 64bc^3a^3m^2 + 48m^5c^3a^2 - 16bc^3a^4m - 96bc^3a^2m^3 \\
& + 64bc^3am^4 + 32m^5bc^3a - 8bc^3a^4m^2 + 32bc^3a^3m^3 + 16m^6c^3 - 48bc^3a^2m^4.
\end{aligned}
$$

$$
\begin{aligned}
D_2^1 = {} & -48bc^2a^3m^3 - 14m^6c^2a^3 - 144bc^2am^5 - 72m^6bac^2 + 12b^2a^3m^3c^2 - 36b^2a^2m^4c^2 \\
& + 24b^2a^3m^2c^2 - 72b^2c^2a^2m^3 - 24bc^2a^3m^4 - 20c^2a^4m^3 + 4c^2a^5m^2 + 10m^4c^2a^4 \\
& + 68c^2a^3m^4 + 96m^6c^2a - 2m^4c^2a^5 + 12m^7c^2a + 24m^7c^2b + 36b^2am^5c^2 \\
& + 72bc^2a^2m^5 - 12m^6b^2c^2 + 6m^7c^2a^2 + 144bc^2a^2m^4 \\
& - 120m^5c^2a^2 + 72b^2c^2am^4 - 8m^8c^2 - 18m^5c^2a^3 - 24b^2m^5c^2 - 28m^7c^2 + 10m^5c^2a^4 \\
& + 6m^6c^2a^2 + 48bc^2m^6 - 2c^2a^5m^3.
\end{aligned}
$$

$$
\begin{aligned}
D_2^2 = {} & -14m^7ca^2 + 4m^8ca + 20m^8c + 6m^7ca^2b - 80m^5ba^2c - 96b^2cam^5 + 16b^3am^5c \\
& - 8m^6ca^3b - 8m^5ca^3b + 8m^7c^3 + 2m^5ca^4b - 16ca^3m^5 + 16ca^3m^4b - 8b^3a^2m^4c \\
& - 16b^3m^5c - 16m^8bc + 24b^2a^2m^5c + 124m^6bca - 4ca^4m^3b - 8m^6b^3c + 32b^3cam^4 \\
& + 48b^2a^2m^4c + 4ca^4m^4 + 24m^7cb^2 - 16b^3a^2m^3c + 8m^6ca^3 - 2ca^4m^5 + 2m^6ba^2c \\
& + 44m^6ca^2 + 2ca^4m^4b - 48b^2am^6c + 20m^7bca - 6m^8ca^2 + 4m^9c - 56m^7bc \\
& - 2m^6ca^4 + 8m^7ca^3 - 52m^7ca + 48b^2m^6c.
\end{aligned}
$$

$$
\begin{aligned}
D_2^3 = {} & -20m^7ba + 2b^4am^5 - 4m^8ba + 4m^9b - 4m^9 - 2m^{10} - 4b^4m^5 + 20m^8b - 16b^3am^5 \\
& + 8m^7b^2a - 8b^3am^6 + 4b^4am^4 + 28m^6b^2a - 2m^6b^4 - 8m^8b^2 + 8m^7b^3 - 28m^7b^2 \\
& + 2m^9a + 16b^3m^6 + 4m^8a.
\end{aligned}
$$